Saif Ullah & Jonathan Masih

Part 1

(1) Macros enable a programmer to extend the assembly language by defining new operations. Macro is a pattern-matching and replacement facility that provides a simple mechanism to name a frequently used sequence of instructions.

(2) A macro performs a task while a subroutine can perform a task with arguments provided and can have a return value. Macro doesn't have a return value and has formal parameters

(3) Assembly language is still important to write programs in which speed or size is critical or to exploit hardware features that have no analogues in high level languages.

(4) Program profiling measures where a program spends its time and can find the time critical parts of a program. It is used to determine which part of the program has a compiling error and needs to be fixed in order for the program to work. Helps work on small sections of program.

(5) 1. In assembly language, the programmer has tight control over which instructions execute. Speed or size of a program is important.
2. The ability to exploit specialized instructions
3. No high-level language is available on a particular computer

6) 1. Programs written in assembly language are inherently machine specific and must totally be rewritten on another computer architecture
2. Assembly language programs are longer than the equivalent programs written in high-level language
3. longer programse are difficult to read/understand and contain more bugs. AL exacerbates the problem because of its complete lack of structure.

7) Forward reference is a label that is used before it is defined. It forces an assembler to translate a program in two steps: first find all labels and then produce instructions.

8) Symbol table is a table that matches names of labels to the addresses of the memory words that instructions occupy. An assembler records the name of the label and the address of the memory word that the instruction occupies. Symbol table records location of each lable defined in the file

9) The three tasks that linkers perform are:
1. Searches the program libraries to find library routines used by the program
2. Determines the memory locations that code from each module will occupy and relocate its instructions by adjusting absolute references
3. Resolves references among files

10) 1. It reads the executable file's header to determine the size of the text and data segments
2. It creates a new address space for the program. This address space is large enough to hold the text and data segments
3. It copies instructions and data from the executable file into

the new address space
4. It copies arguments passed to the program onto the stack
5. It initializes the machine registers. In general, most registeres are cleared, but the stack pointer must be assigned the address of the first free stack location
6. It jumps to a start-up routine that copies the program's arguments from the stack to registers and calls the program's main routine. If the main routine returns, the start up routine terminates the program with the exit system call.